Open in app ↗

Search Medium

7

Model Diagnostics Trilogy — Part 1

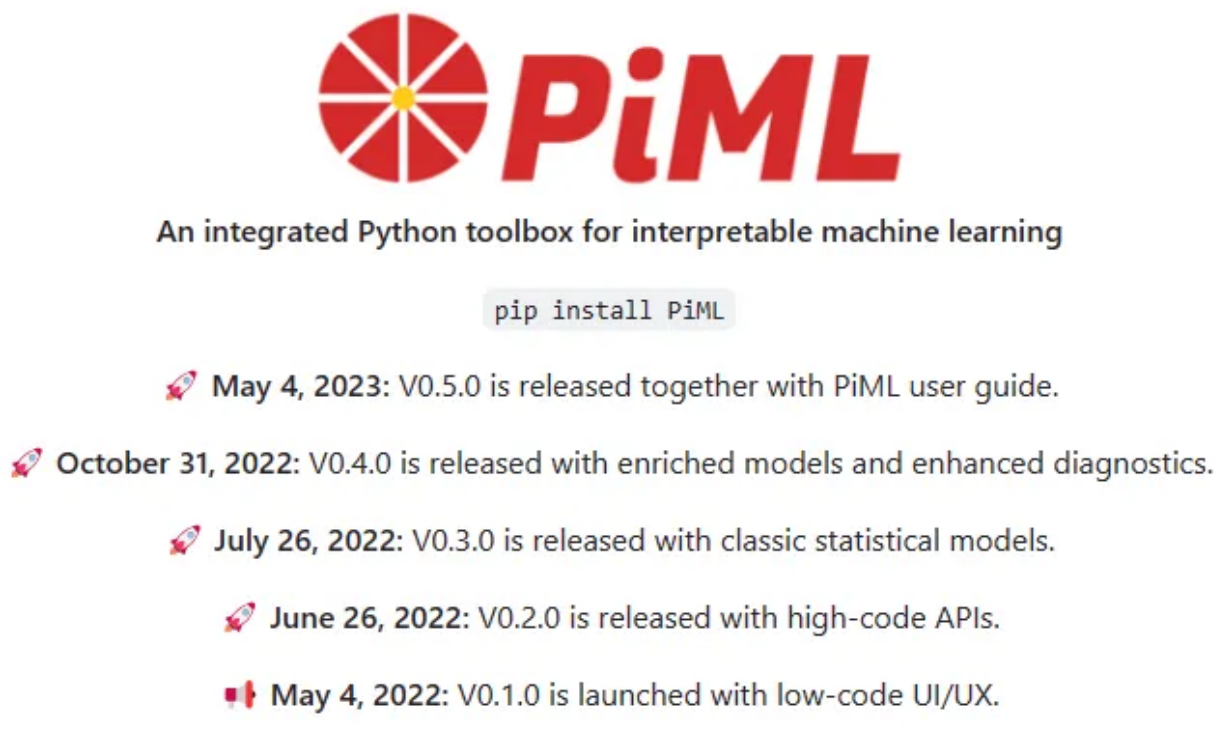
Error Analysis and Resilience Test

PiML Tutorials · Follow

7 min read · 15 hours ago

Listen Share More

> *PiML was born as an educational toolbox for interpretable machine learning in Python community. It is designed for both model development and model diagnostics. Since its first launch on May 4th, 2022, we a small development team have made several major updates with expanding features and capabilities, together with a complete* ***user guide****.*


PiML

An integrated Python toolbox for interpretable machine learning

`pip install PiML`

🚀 **May 4, 2023:** V0.5.0 is released together with PiML user guide.

🚀 **October 31, 2022:** V0.4.0 is released with enriched models and enhanced diagnostics.

🚀 **July 26, 2022:** V0.3.0 is released with classic statistical models.

🚀 **June 26, 2022:** V0.2.0 is released with high-code APIs.

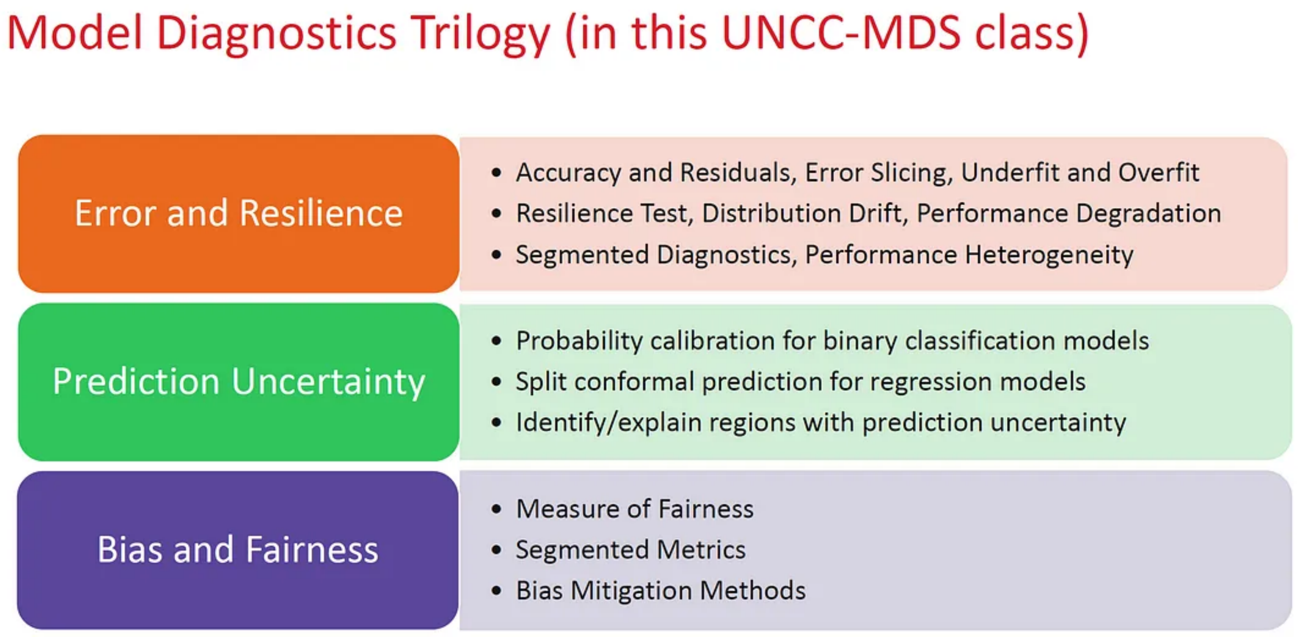
📢 **May 4, 2022:** V0.1.0 is launched with low-code UI/UX.

Source: https://github.com/SelfExplainML/PiML-Toolbox/

> *We've recently made a minor update to its latest version 0.5.1, incorporating features based on user request through* ***Github issues****.*

```
Sep 28, 2023 PiML v0.5.1 release with new features:
- Model-free diagnostic test APIs.
- Sliced overfitting test with ACC and AUC metrics.
- Other miscellaneous improvements and bug fixes.
- Support for a wider range of computing environments, including Mac ARM.
```

> *As it happens, Aijun from the PiML Team is involved in teaching a model validation class for Master of Data Science students at UNC Charlotte in Fall 2023. Three topics of model diagnostics from the syllabus will be covered:*


Model Diagnostics Trilogy (in this UNCC-MDS class)
Error and Resilience
• Accuracy and Residuals, Error Slicing, Underfit and Overfit
• Resilience Test, Distribution Drift, Performance Degradation
• Segmented Diagnostics, Performance Heterogeneity
Prediction Uncertainty
• Probability calibration for binary classification models
• Split conformal prediction for regression models
• Identify/explain regions with prediction uncertainty
Bias and Fairness
• Measure of Fairness
• Segmented Metrics
• Bias Mitigation Methods

Topics of Model Diagnostics, UNCC Master of Data Science, Fall 2023

> *We convert Aijun's lecture notes on these topics to PiML tutorials, together with case studies and Python codes that are shared* ***here****.*

Machine Learning Model Diagnostics

🏆**Model performance is not all you need.**

ML model performance is often measured by **accuracy**, as examined via standard overall metrics (e.g. MSE, MAE, R2, ACC, AUC, F1-score, Precision and Recall).

However, model risk assessment by single-valued metrics is insufficient. More granular diagnostics and outcome testing are needed, including

- **Resilience test:** anticipate performance degradation due to input distribution drift
- **Reliability test:** assess prediction confidence by uncertainty quantification
- **Robustness test:** assess performance degradation due to small input perturbation

A primary goal of model diagnostics is **weakness identification**, i.e., to identify regions and drivers where the model performs weak in various ways. The model may have an overall performance, but may be underfitting or overfitting in sub regions, due to under/overrepresenting local nonlinearity or interaction effects. It may also be exposed to unreliable regions with larger prediction uncertainty. Sometimes these model weaknesses can be detectable, explainable and fixable. In other times the model may be revealed with potential limitation due to sparse data or low signal-to-noise ratio, thus facing a difficult-to-predict problem.

Bias and fairness is another important aspect of model diagnostics, for ensuring responsible practice of machine learning. It is to identify disparity or discrimination against demographic groups and mitigate bias.

All of these diagnostic tests are covered by the PiML toolbox. In this tutorial, we start with error analysis and resilience test.

Part 1: Error and Resilience

Suppose we are given two example models, one is a regression case for the California Housing data, the other a binary classification case for the SimuCredit data. The Python notebooks can be found from PiML Github or experimented directly through Google Colab:

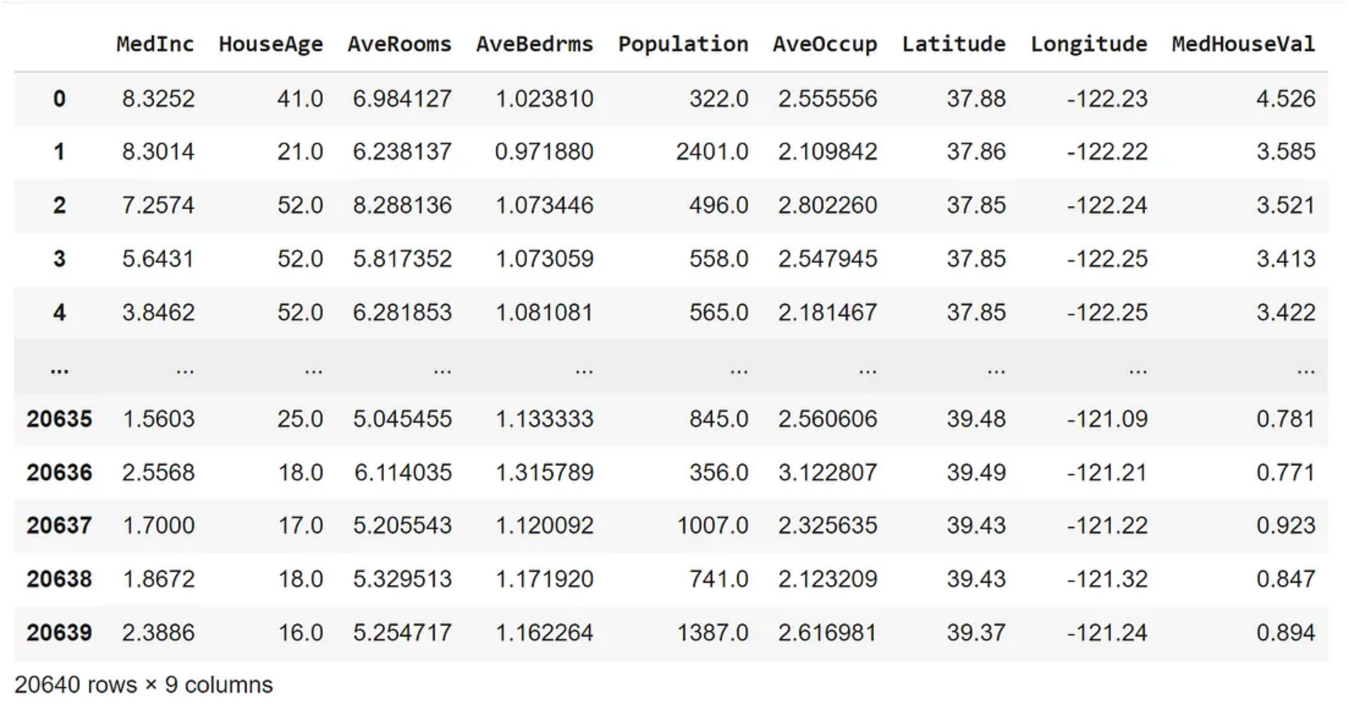
1. CaliforniaHousing Case (Regression)
2. SimuCredit Case (Binary Classification)

In what follows, we focus on the regression case.

🤖Data and Model Pipelines

PiML provides the convenient low-code interfaces for data, model and test pipelines. After installing the package by “*pip install piml*”, one may initiate an experiment and load data:

```
from piml import Experiment
exp = Experiment()
exp.data_loader(data='CaliforniaHousing_trim2')
```

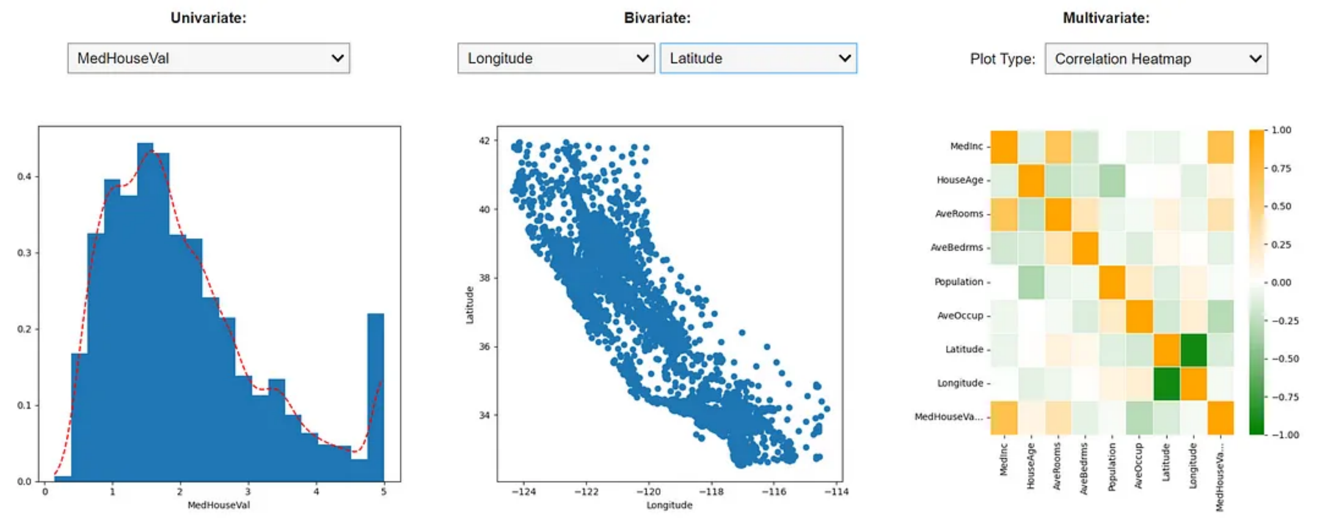


	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422
...	...	...	...	...	...	...	...	...	...
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09	0.781
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21	0.771
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22	0.923
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32	0.847
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24	0.894

20640 rows × 9 columns

California Housing data (trim2 version in PiML Toolbox)

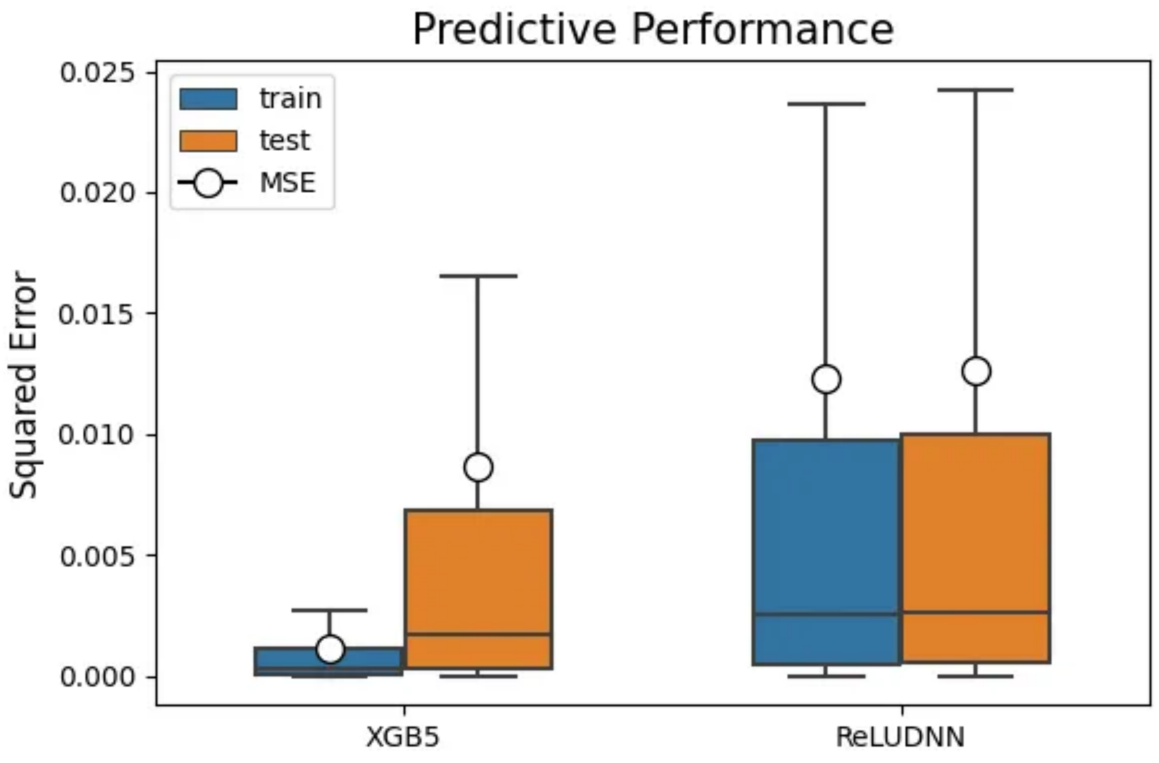
Data summary, preprocessing and visualization can be easily carried out in PiML. The last variable *MedHouseVal* is the response; as usual, we split the data into 80–20 for training and testing. Please refer to the example notebook.



In model pipeline, PiML provides a whole list of interpretable machine learning models that come with model-specific interpretation, as well as post-hoc explanation. We will have a dedicated thread covering them; stay tuned.

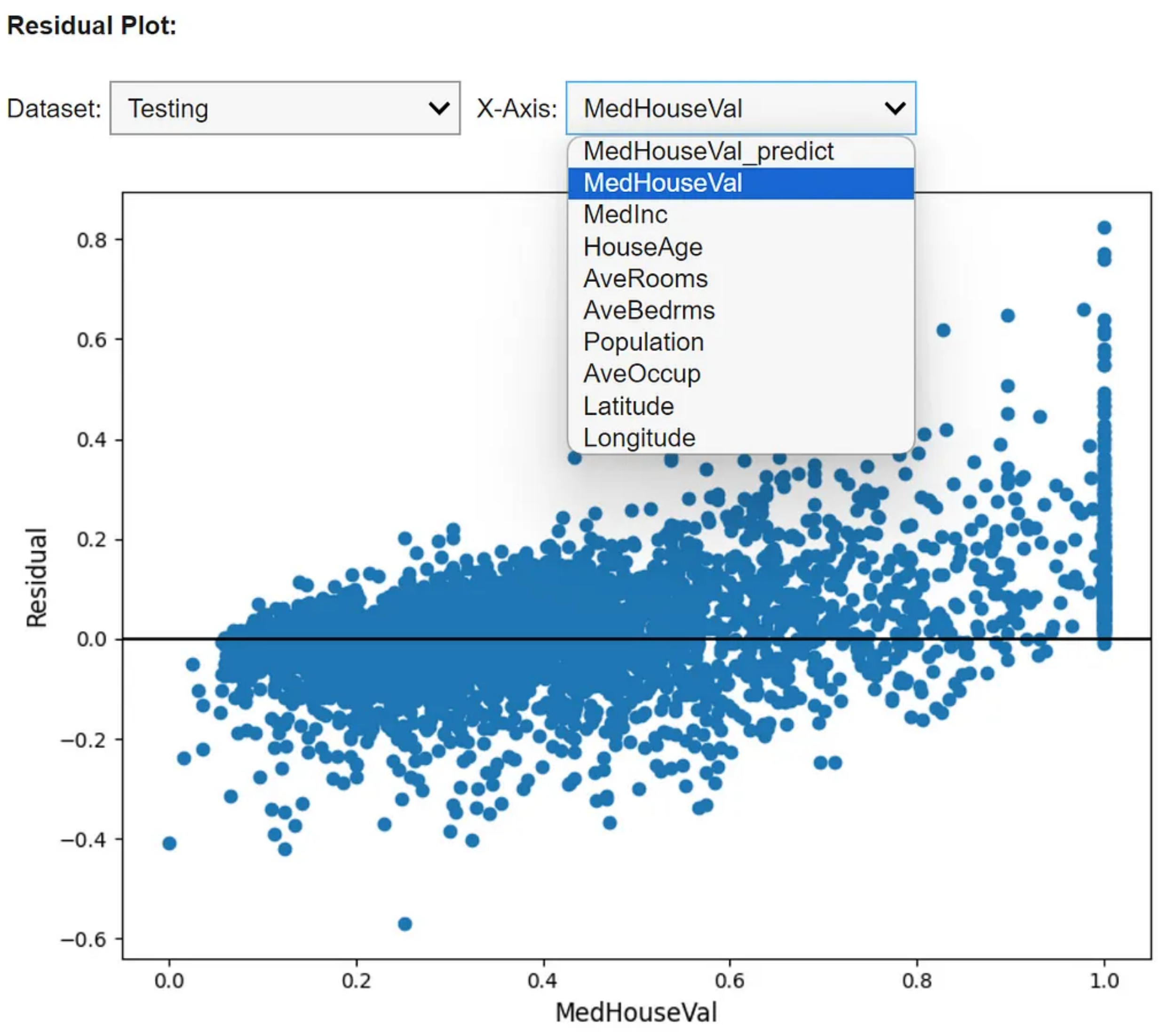
In this tutorial, let's take XGBoost (depth 5) and DNN (ReLU activation) off the shelf, register them into PiML pipeline, and check their prediction accuracy. It is shown that XGB5 performs better than ReLU-DNN, even when XGB5 is overfitting.

```
from xgboost import XGBRegressor
XGB = XGBRegressor(max_depth=5, n_estimators=500)
exp.model_train(model=XGB, name='XGB5')

from sklearn.neural_network import MLPRegressor
DNN = MLPRegressor(hidden_layer_sizes=[40]*4,
                   activation="relu", random_state=0)
exp.model_train(model=DNN, name='ReLUDNN')

exp.model_compare(models=["XGB5", "ReLUDNN"],
                  show="accuracy_plot", metric="MSE",
                  figsize=(6, 4))
```



🔬Diagnostics by Residual Plot

Let’s diagnose ReLU-DNN by PiML ***exp.model_diagnose()***. The accuracy tab shows not only performance metrics like MSE, MAE and R2, but also the residual plot where the user can choose training/testing data and x-axis. From the result below, it is evident that the model does not perform well on large response values.



🔬Diagnostics by Error Slicing

To pinpoint model weaknesses at a granular level, PiML offers error slicing techniques.

1. **Specify an appropriate metric** based on individual prediction residuals: e.g., MSE for regression, ACC for classification, train-test performance gap, prediction interval bandwidth, etc.

2. **Specify 1 or 2 slicing features** of interest;

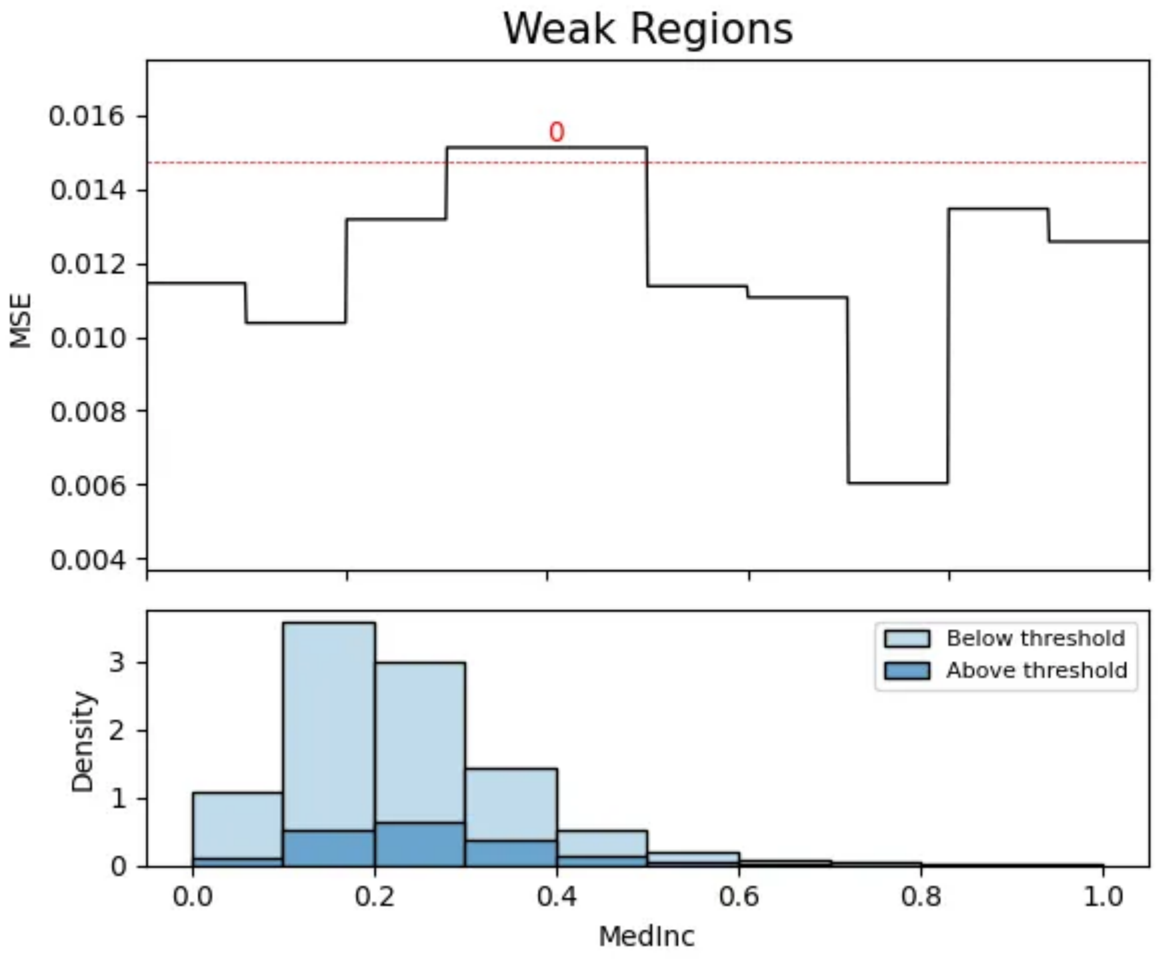
3. **Evaluate the metric** for each sample in the target data (training or testing) as pseudo responses;

4. **Segment the target data** along the slicing features, by a) Histogram slicing with equal-space binning, or b) fitting a decision tree or tree-ensemble to generate the sub-regions;

5. **Identify the sub-regions** with average metric exceeding the pre-specified threshold, subject to minimum sample condition.

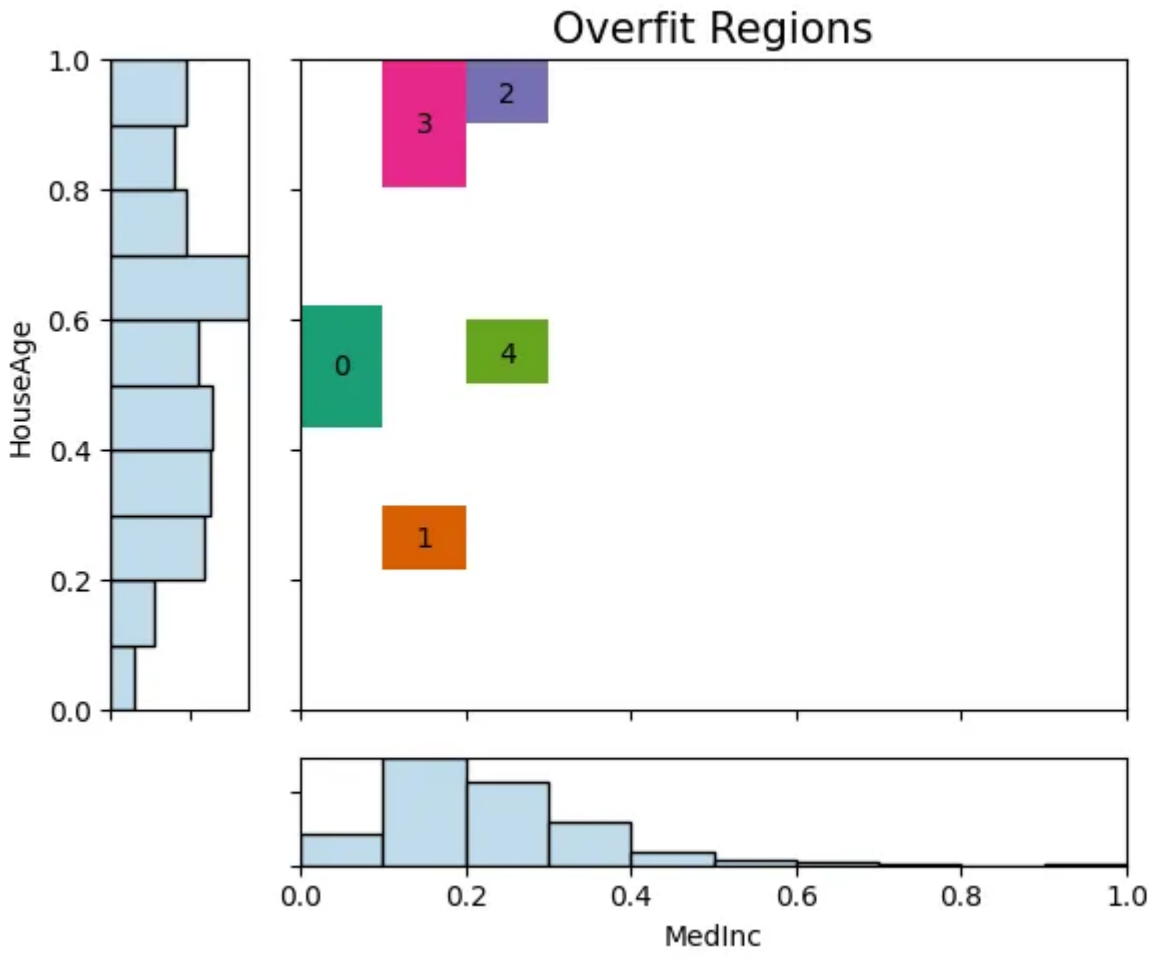
Slicing can be applied to the training data for check of underfitting regions, which is called WeakSpot in PiML.

```
exp.model_diagnose(model="ReLUDNN", show="weakspot", metric="MSE",
          slice_method="histogram", slice_features=["MedInc"],
          threshold=1.2, min_samples=20, use_test=False, figsize=(6,5))
```



Slicing can be also applied with train-test performance gap for check of overfitting regions:

```
exp.model_diagnose(model="ReLUDNN", show="overfit", metric="MSE",
                   slice_method="histogram", slice_features=["MedInc", "HouseAge"],
                   threshold=1.2, min_samples=100, figsize=(6, 5))
```



🚀Resilience Test

Resilience test is to anticipate performance degradation under covariate distribution drift. Distributionally resilient models would show mild degradation in performance under distribution drift, under the risk minimization framework:

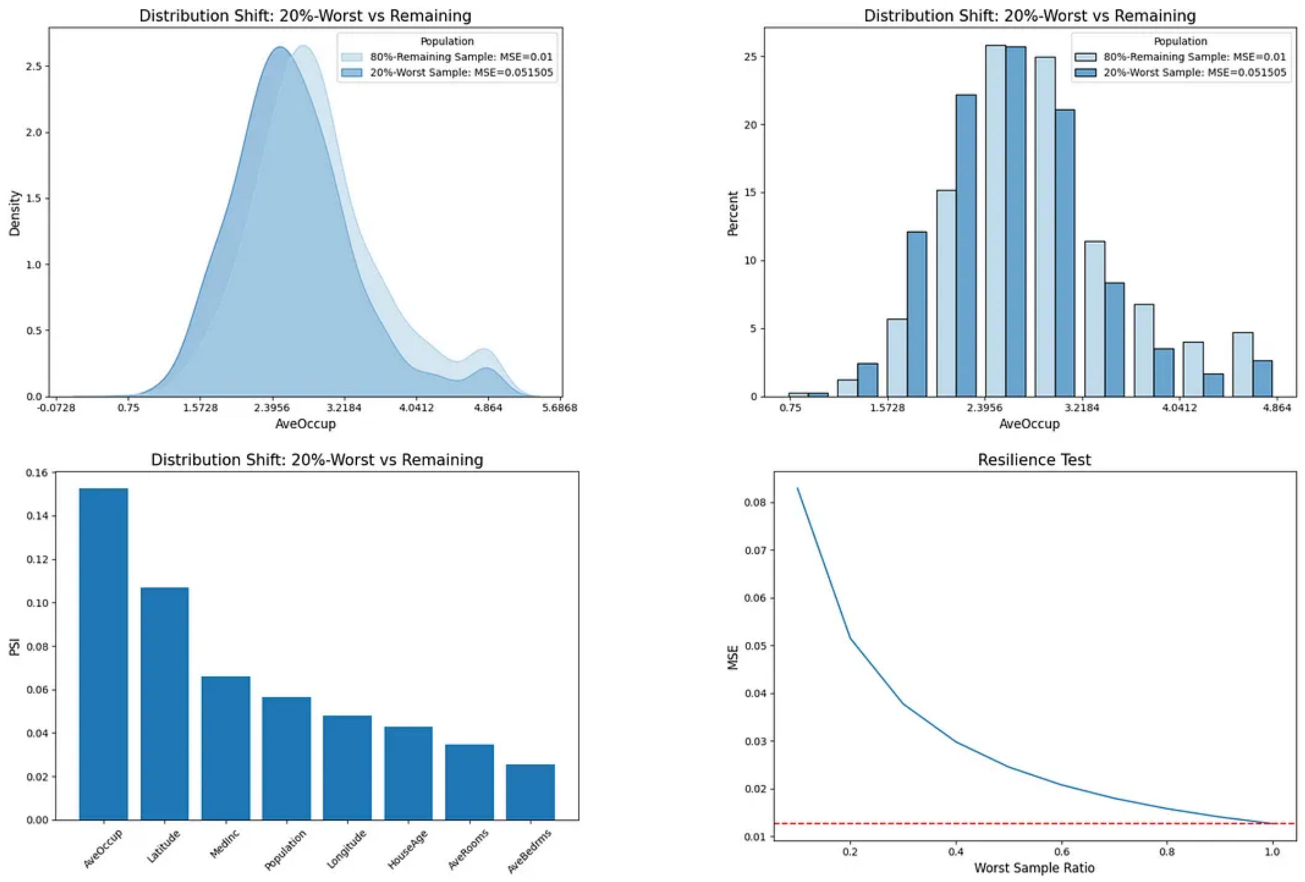
$$\min_{\theta\in\Omega} \max_{Q\in\mathcal{S}} \mathbb{E}_Q\left[l(X,Y;\theta)\right]$$

where *S* represents the set of distribution drift scenarios. There are four resilient scenarios offered in the PiML toolbox:

1. **Worst-sample:** percentage of worst-performing test samples based on residual magnitude
2. **Worst-cluster:** worst-performing cluster of test samples based on K-means clustering
3. **Outer-sample:** percentage of boundary/outlying test samples distant from the sample mean
4. **Hard-sample:** percentage of difficult-to-predict test samples based on an auxiliary model

Note that scenarios 1 and 2 are model-specific, while scenarios 3 and 4 are model-agnostic.

Run PiML ***exp.model_diagnose()*** for ReLU-DNN and choose the resilience tab. Let's illustrate the resilience test with worst-sample scenario. The last plot shows the curve of performance degradation as the percentage of worst-performing test samples varies. For a user-defined ratio (here 20%), it also shows the distribution shift between 20%-worst samples versus the remaining 80% samples.

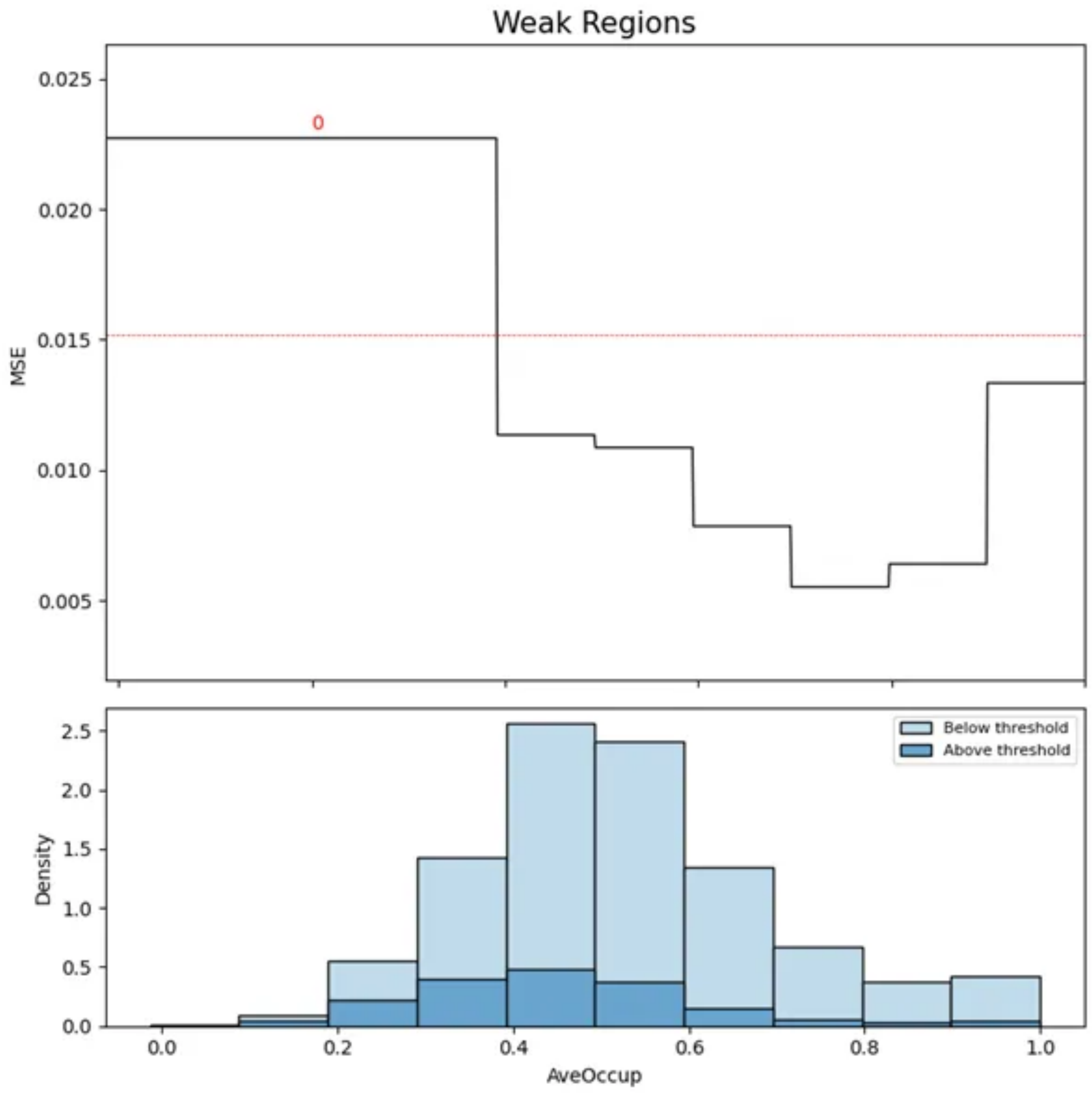


The distribution drift can be measured by two-sample test between empirical distributions, including Kullback-Leibler (KL) divergence, Kolmogorov-Smirnov (KS) and Cramer-von Mises (CM) statistics, Population Stability Index (PSI), and Wasserstein distance. Among others, PSI tests one variable at a time,

$$PSI = \sum_{i=1}^{B} (\text{Target}_i\% - \text{Base}_i\%) \ln\left(\frac{\text{Target}_i\%}{\text{Base}_i\%}\right)$$

based on the proportions of samples in each bucket of the target vs. base population. As a rule of thumb, when PSI ≥ 0.2, it indicates a significant distribution change. PSI between 0.1 and 0.2 indicates a modest change. The variables with notable drift are deemed to be sensitive or vulnerable in the resilience test.

From the plots above, the variable *AveOccup* in CaliforniaHousing data is checked with modest distribution drift. Its sensitivity to worst-performing samples can be verified through PiML WeakSpot over testing samples, where the smaller *AvgOccup* regions tend to be more difficult to predict than the larger counterpart.

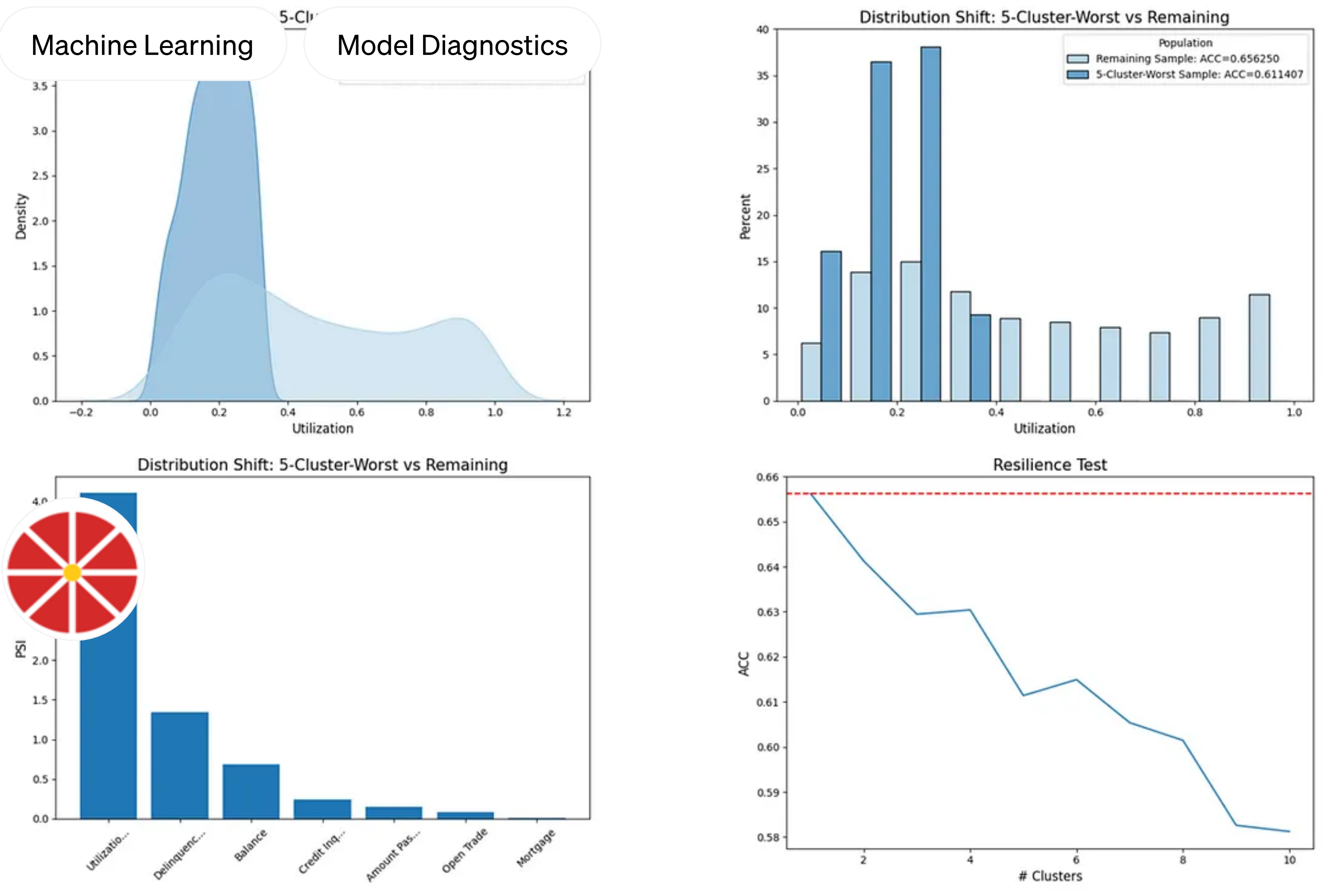


⛳**One More Thing**

In the resilience test, we may choose other distribution drift scenarios, e.g. worst-cluster for the binary classification model XGB5 based on SimuCredit data, with result shown below.

Machine Learning

Model Diagnostics



It is found that a sub-region of *Utilization* is tied with the worst-performing cluster among K-Means clusters with K = 5.

Recommended from Medium

Following this spirit, we can conduct model diagnostics on a segment-by-segment basis. In the supplementary Python notebooks for both cases, an experimental segmented diagnostics was coded to check performance heterogeneity, by utilizing the latest released PiML scored_test APIs:

```
from piml.scored_test import test_accuracy, residual_plot, slicing_weakspot
```

We would invite you to join discussion about best practices for segmented diagnostics. We wish to roll it out as a new feature in future PiML release.

Thank you for reading!